xCAT 2.0 Monitoring Howto

5/29/2008

Table of Contents

1.0 Introduction

There are two monitoring infrastructures introduced in xCAT 2.0. The ***xCAT Monitoring Plug-in Infrastructure*** allows you to plug-in one or more third party monitoring software such as Ganglia, RMC, SNMP etc. to monitor the xCAT cluster. The ***xCAT Notification Infrastructure*** allows you to watch for the changes in xCAT database tables.

2.0 Using xCAT Monitoring Plug-in Infrastructure

With xCAT 2.0, you can integrate 3rd party monitoring software into your xCAT cluster. The idea is to use monitoring plug-in modules that act as bridges to connect xCAT and the 3rd party software. Though you can write your own monitoring plug-in modules (see section 2.3), over the time, xCAT will supply a list of built-in plug-in modules for the most common monitoring software. They are:

- xCAT (xcatmon.pm) (monitoring node statue using fping. released)
- SNMP (snmpmon.pm) (snmp monitoring. released)
- RMC (rmcmon.pm)
- Ganglia (gangliamon.pm)
- Nagios (nagiosmon.pm)

You can pick one or more monitoring plug-ins to monitor the xCAT cluster. The following sections will demonstrate how to use snmpmon and xcatmon plug-ins.

2.1 Define monitoring servers

You can skip this section if you have a small number of nodes to monitor, or if you prefer the management node (mn) handles the monitoring loads. For a large cluster, it is

recommended that you dedicate some nodes as monitoring aggregation points. These nodes are called monitoring servers. You can use the service nodes (sn) as the monitoring servers. The monitoring servers are defined by the 'monserver' column of the **noderes** table. The data in 'monserver' column is a comma separated pairs of host names or ip addresses. The first host name or ip address represents the network adapter on that connects to the mn. The second host name or ip address represents the network adapter that connects to the nodes. If the no data is provided in the 'monserver' column, the values in the 'servicenode' and the 'xcatmaster' columns in the same table will be used. If none is defined, the mn will be used as the monitoring server.

In following example the nodes in group2 have dedicated monitoring server (monsv02) while the nodes in group1 use their service node as the monitoring server (sn01).

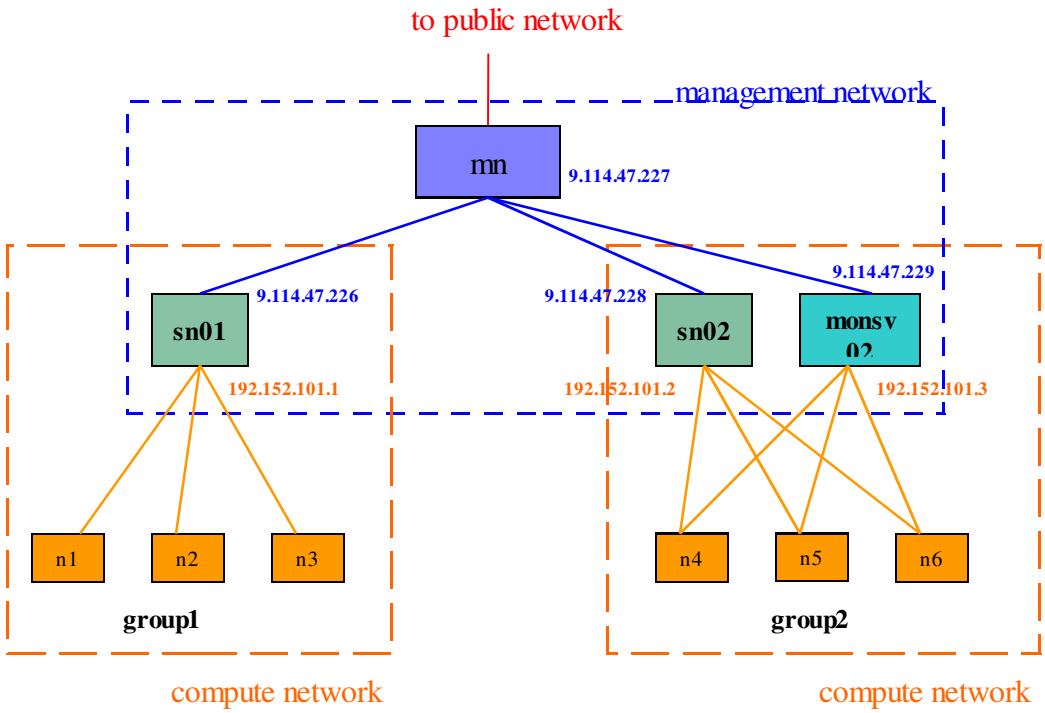

Figure 1. Monitoring servers for the nodes

The **noderes** table looks like this for the above cluster.

node	monservers	servicenode	xcatmaster
sv01		9.114.47.227	9.114.47.227
sv02		9.114.47.227	9.114.47.227
monsv02		9.114.47.227	9.114.47.227
group1		sv01	192.152.101.1
group2	monsv02, 192.152.101.3	sv02	192.152.101.2

2.2 Enable SNMP monitoring

1. Down load the corresponding mib files for your system that you wish to receive SNMP traps from and copy them onto the management node (mn) and the monitoring servers under the following directory:

```
/usr/share/snmp/mibs/
```

The mib files for IBM blade center management modules (MM) and RSAII are packaged within the firmware updates. They can be found under IBM support page:

http://www.ibm.com/support/us/en/

For example,

Download mibs for MM:

- Go to http://www-304.ibm.com/systems/support/supportsite.wss/docdisplay?lndocid=MIGR-5070708&brandind=5000020
- Download file `ibm_fw_amm_bpet26k_anyos_noarch.zip`
- Unzip the file and you will fine two mib files `mmblade.mib` and `mmalert.mib`

Down load mibs for RSAII

- Go to http://www-304.ibm.com/systems/support/supportsite.wss/docdisplay?brandind=5000008&lndocid=MIGR-64575
- Download file `ibm_fw_rsa2_ggep30a_anyos_noarch.zip`
- Unzip it and you will find the mib files `RTRSAAG.MIB` and `RTALSERT.MIB`

2. Make sure net-snmp rpm is installed on mn and all the monitoring servers.

```
rpm -qa |grep net-snmp
```

3. Start the monitoring

```
monstart snmpmon
```

4. Set email recipients

When traps are received, they will be logged into the syslog on mn. The warning and critical alerts will be emailed to 'alserts' alias on the mn's mail system. By default, 'alerts' points to the root's mailbox, but you can have the emails sent to other recipients by modifying it. On mn

```
vi /etc/aliases
```

Fine the line beginning with the word `alerts`; it usually is at the bottom of the file. Change the line so it looks something like this

```
alerts    root,joe@us.ibm.com,jill@yahoo.com
```

Now make the new email aliases in effect

```
newaliases
```

5. Set up the filters

xCAT built-in SNMP trap handler can process any SNMP traps. Here is a sample email message sent by the trap handler after a Blade Center MM trap is handled.

```
Subject: Critical: Cluster SNMP Alert!
Message:
  Node: rro123b
  Machine Type/Model:  0284
  Serial Number:  1012ADA
  Room:
```

```
  Rack:
  Unit:
  Chassis:
  Slot:

SNMP Critical Alert received from bco41(UDP: [11.16.15.41]:161)
  App ID: "BladeCenter Advanced Management Module"
  App Alert Type: 128
  Message: "Processor 2 (CPU 2 Status) internal error"
  Blade Name: "rro123b"
  Error Source="Blade_11"

Trap details:
    DISMAN-EVENT-MIB::sysUpTimeInstance=17:17:49:12.08
    SNMPv2-MIB::snmpTrapOID.0=BLADESPPALT-MIB::mmTrapBladeC
    BLADESPPALT-MIB::spTrapDateTime="Date(m/d/y)=05/20/08, Time(h:m:s)=14:30:12"
    BLADESPPALT-MIB::spTrapAppId="BladeCenter Advanced Management Module"
    BLADESPPALT-MIB::spTrapSpTxtId="bco41"
    BLADESPPALT-MIB::spTrapSysUuid="D76ADB0137E2438B9F14DCC6569478BA"
    BLADESPPALT-MIB::spTrapSysSern="100058A"
    BLADESPPALT-MIB::spTrapAppType=128
    BLADESPPALT-MIB::spTrapPriority=0
    BLADESPPALT-MIB::spTrapMsgText="Processor 2 (CPU 2 Status) internal error"
    BLADESPPALT-MIB::spTrapHostContact="No Contact Configured"
    BLADESPPALT-MIB::spTrapHostLocation="No Location Configured"
    BLADESPPALT-MIB::spTrapBladeName="rro123b"
    BLADESPPALT-MIB::spTrapBladeSern="YL113684L129"
    BLADESPPALT-MIB::spTrapBladeUuid="3A77351D00001000B6AA001A640F4972"
    BLADESPPALT-MIB::spTrapEvtName=2154758151
    BLADESPPALT-MIB::spTrapSourceId="Blade_11"
    SNMP-COMMUNITY-MIB::snmpTrapAddress.0=11.16.15.41
    SNMP-COMMUNITY-MIB::snmpTrapCommunity.0="public"
    SNMPv2-MIB::snmpTrapEnterprise.0=BLADESPPALT-MIB::mmRemoteSupTrapMIB
```

But sometimes you want the trap handler filter out certain type of alerts. For example, when blades are rebooting you will get a lot of alerts and you do not want to be notified for these alerts. The filtering can be done by adding a row in the '**monsetting**' table with name equals to `snmpmon` and key equals to `ignore`. The value is a comma separated list that describes the contents in a trap.

For example, to filter out any blade center mm traps from blade rro123b.

```
chtab name=snmpmon,key=ignore monsetting.value=BLADESPPALT-MIB::spTrapBladeName="rro123b"
```

(The mib module name `BLADESPPALT-MIB` is optional in the command. `spTrapBladeName` can be found in the mm mib file or from your email notification.)

The following example will filter out all power on/off/reboot alerts for any blades.

```
chtab name=snmpmon,key=ignore monsetting.value=spTrapMsgText=”Blade powered off”,spTrapMsgText=”Blade powered on”,spTrapMsgText=”System board (Sys Pwr Monitor) power cycle”,spTrapMsgText=”System board (Sys Pwr Monitor) power off”,spTrapMsgText=”System board (Sys Pwr Monitor) power on”,spTrapMsgText=”Blade reboot”
```

There are other keys and values for the **monsetting** table supported by snmpmon monitoring plug-in. For example, you can make user-defined commands to be run for certain traps by adding '`runcmd`' key in the table. Use this command to list all the possible keywords.

```
monls snmpon -d
```

6. Make sure the blade names on Blade Center MM are identical to the node names defined in the xCAT nodelist table.

```
rspconfig group1 textid        (This command queries the blade name)
  n1: textid: SN#YL10338241EA
  n2: textid: SN#YL103382513F
  n3: textid: SN#YK13A084307Y
rspconfig group1 textid=*      (This command sets the blade name)
  n1: textid: n1
  n2: textid: n2
  n3: textid: n3
```

7. Verify

Make sure snmpmon is activated.

```
monls snmpmon
  snmpmon       monitored
```

Make sure snmptrapd is up and running on mn and all monitoring servers. And it has `-m ALL` flag.

```
ps -ef |grep snmptrapd
  root    31866    1  0 08:44 ?       00:00:00 /usr/sbin/snmptrapd -m ALL
```

Make sure snmp destination is set to the corresponding monitoring servers.

`rspconfig mm snmpdest` (mm is the group name for all the blade center management modules)

```
  mm1: SP SNMP Destination 1:  192.152.101.1
  mm2: SP SNMP Destination 1:  192.152.101.3
```

Make sure SNMP alert is set to 'enable'

```
rspconfig mm alert
  mm1: SP Alerting: enabled
  mm2: SP Alerting: enabled
```

(Use `monstop snmpmon` to stop the monitoring if desired.)

***Note: Please stop snmpmon before restart xcatd.**

If at some stage you decide to restart xcatd, you must stop snmpmon first. This is because restarting xcatd stops and restarts snmpmon. snmpmon starting and stopping processes take long time and they are running asynchronously with xcatd and with each other. SNMP monitoring may not be initialized correctly if you do not give it enough time. Here is the procedure to restart xcatd when snmpmon is active:

```
monstop snmpmon
service xcatd restart
monstart snmpmon
```

2.3 Enable node liveness monitoring

xcatmon provides node liveness monitoring using fping. This can be used if no other 3rd party software is used for node status monitoring. The ***status*** column of the ***nodelist*** table will be updated periodically with the latest node liveness status by this plug-in.

1. To activate, use the monstart command:

```
monstart xcatmon -n -s [ping-interval=5]
```

where 2 means that the nodes are pinged for status every 2 minutes**.**

2. Verify

Make sure xcatmon is activated.

```
monls snmpmon
  xcatmon        monitored   node-status-monitored
```

Check the setting

```
tabdump monsetting
  #name,key,value,comments,disable
  "xcatmon","ping-interval","5",,
```

Make sure cron jobs are activated on mn and all monitoring server

```
crontab -l
  */5 * * * * XCATROOT=/opt/xcat PATH=/bin:/usr/bin:/sbin:/usr/sbin:/opt/xcat/bin:/opt/xcat/sbin /opt/xcat/sbin/xcatnodemon
```

2.4 Create your own monitoring plug-in module

As mentioned before, a monitoring plug-in modules acts as a bridge to connect xCAT and the 3rd party software. The functions of a monitoring plug-in module include initializing the 3rd party software, informing it with the changes of the xCAT node list, setting it up to feed node status back to xCAT etc. The following figure depicts the data flow and the relationship among xcatd, monitoring plug-ins and the third party software.

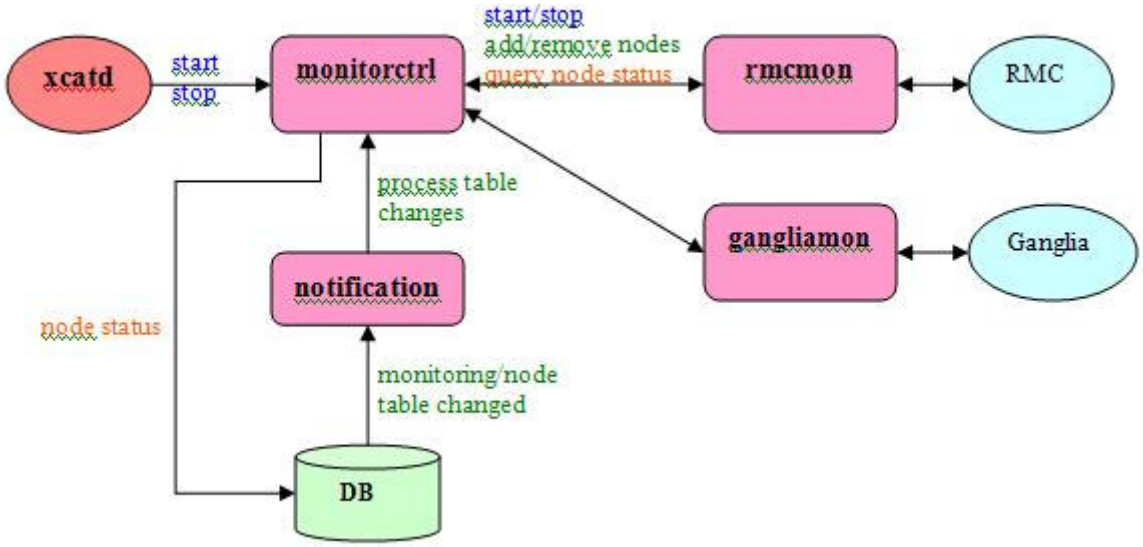

Figure 2. Data flow among xcatd, plug-in modules and 3rd party monitoring software

To use this infrastructure to create your own plug-in module, create a Perl module and put it under **`/opt/xcat/lib/perl/xCAT_monitoring/`** directory. If the file name is xxx.pm then the package name will be `xCAT_monitoring::xxx`. The following is a list of subroutines that a plug-in module must implement:

start

stop

supportNodeStatusMon

startNodeStatusMon

stopNodeStatusMon

addNodes

removeNodes

processSettingChanges

getDiscription

Please refer to */opt/xcat/lib/perl/xCAT_monitoring/samples/tmplatemon.pm* for the detailed description of the functions. You can find *tmplatemon.pm* from the xCAT source code on the web: http://xcat.svn.sourceforge.net/viewvc/xcat/xcat-core/trunk/xCAT-server-2.0/lib/xcat/monitoring/samples/

To activate the monitoring with xxx, use the following command:

```
monstart xxx [-n|--nodestatmon] [-s|--settings settings]
```

where

-n or --nodestatmon indicates it can help feeding the node status to xCAT. The node status is stored in the ***status*** column of the ***nodelist*** table.

-s or –settings specifies the plug-in specific settings. These setting will be used by the plug-in to customize certain entities for the plug-in or the third party monitoring software. The format of the setting string is: [key=value],[key=value]… Please note that the square brackets are needed. e.g. [mon_interval=10],[toggle=1]

Example**:**

```
monstart xxx -n
```
(with feeding the node status to xCAT table)

or

```
monstart xxx
```
(not feeding the node status to xCAT table)

Once it is activated, xCAT will automatically, through the plug-in module, start the 3rd party software for monitoring. To deactivate the monitoring plug-in and stop the monitoring use this command:

```
monstop xxx
```

3.0 Using xCAT Notification Infrastructure

With xCAT 2.0, you can monitor xCAT database for changes such as nodes entering/leaving the cluster, hardware updates, node liveness etc. In fact anything stored in the xCAT database tables can be monitored through the xCAT notification infrastructure. To start getting notified for changes, simply register your Perl module or command as the following:

```
regnotif filename tablename -o actions
```

where

filename is the full path name of your Perl module or command.

tablenames is a comma separated list of table names that you are interested in.

actions is a comma separated list of data table actions. 'a' for row addition, 'd' for row deletion and 'u' for row update.

Example**:**

```
regnotif /opt/xcat/lib/perl/xCAT_monitoring/mycode.pm nodelist,nodhm
-o a,d
```

```
regnotif /usr/bin/mycmd switch,noderes -o u
```

Use the following command to view all the modules and commands registered.

```
tabdump notification
```

To unregister, just do the following:

```
unregnotif filename
```

Example:

```
unregnotif  /opt/xcat/lib/perl/xCAT_monitoring/mycode.pm
unregnotif  /usr/bin/mycmd
```

If the *filename* specifies a Perl module, the package name must be **xCAT_monitoring::xxx**. It must implement the following subroutine which will get called when database table change occurs:

processTableChanges*(tableop, table_name, old_data, new_data)*

where:

tableop Table operation. It can be 'a' for row addition, 'd' for row deletion and 'u' for row update.

tablename The name of the database table whose data has been changed.

old_data An array reference of the old row data that has been changed. The first element is an array reference that contains the column names. The rest of the elements are array references each contains attribute values of a row. It is set when the action is u or d.

new_data A hash reference of the new row data; only changed values are in the hash. It is keyed by column names. It is set when the action is u or a.

If the file name specifies a command (written by any programming languages or scripts), when the interested database table changes, the info will be fed to the command through the standard input. The format of the data in the STDIN is as following:

```
action(a, u or d)
tablename
[old value]
col1_name,col2_name...
col1_val,col2_val,...
col1_val,col2_val,....
...
[new value]
col1_name,col2_name,...
col1_value,col2_value,...
...
```

The sample code can be found under `/opt/xcat/lib/perl/xCAT_monitoring/samples/mycode.pm` on a installed system or on the web http://xcat.svn.sourceforge.net/viewvc/xcat/xcat-core/trunk/xCAT-server-2.0/lib/xcat/monitoring/samples/